Mario R. Eden, Marianthi Ierapetritou and Gavin P. Towler (Editors) Proceedings of the 13th International Symposium on Process Systems Engineering – PSE 2018
July 1-5, 2018, San Diego, California, USA © 2018 Elsevier B.V. All rights reserved.

Expanding RTN discrete-time scheduling formulations to preemptive tasks

Pedro M. Castro[a*], Iiro Harjunkoski[b], Ignacio E. Grossmann[c]

[a]*CMAFCIO, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal*
[b]*ABB AG, Corporate Research Center, Ladenburg, Germany*
[c]*Dep. Chemical Engineering, Carnegie Mellon University, Pittsburgh PA, USA*
pmcastro@fc.ul.pt

Abstract

This paper expands the Resource-Task Network (RTN) scheduling formulation to allow tasks to be interrupted when encountering a planned break period in production. The benefit from a more flexible mode of operation, is the improvement of overall equipment efficiency. This is illustrated by solving a benchmark problem from the literature. To address one of the major limitations of discrete-time approaches, we revisit solution strategies for the objective of makespan minimization, before proposing a method to reduce the number of iterations in the search for the optimal solution.

Keywords: Optimization; Mixed-integer linear programming; Algorithms.

1. Introduction

Enterprise-wide optimization (Grossmann, 2005) aims to simultaneously account for key performance indicators across multiple business units by looking into the integration of supply chain management, production control, planning and scheduling. To accomplish this goal, we need to be able to transfer data and information efficiently between different industrial production management systems. The ISA-95 standard can act as a data-exchange platform for production scheduling (Harjunkoski and Bauer, 2014) but it needs to be linked to a scheduling formulation that can cope with the wide variety of features that may be encountered at a process plant.

The most general scheduling formulations, based on unified frameworks for process representation (State-Task and Resource-Task Network), lack the possibility of processing tasks to be interrupted. Yet, preemption is often encountered in daily practice, with the ISA-95 input including planned break periods (e.g., due to preventive maintenance, weekends), number of stops allowed per task, minimum duration of each active partial task and possible penalties for interrupting a task.

Models that allow for preemption exist in the context of project scheduling. For instance, van Peteghem and Vanhoucke (2010) assume that an activity with discrete duration τ can be interrupted up to τ times, with the time(s) of interruption being determined by a genetic algorithm. In this paper, we consider a more constrained for of preemption, where tasks can be interrupted at given points in time, provided that they continue immediately after. We focus on the RTN discrete-time formulation (Pantelides, 1994), which is known to be very tight and better than continuous-time formulations at handling discrete events (Harjunkoski et al. 2014). Interestingly, the required changes occur at the level of the structural parameters and excess resource balance constraints, with the model variables

remaining the same. It is straightforward to extend a STN discrete-time formulation (Kondili et al., 1993; Lee and Maravelias, 2017) to handle preemption in a similar way.

2. Preemptive vs. non-preemptive scheduling

Non-preemptive scheduling is the standard mode of operation in PSE models, see Figure 1. Given a break period br occurring in time window $[b_{br}^{L}, b_{br}^{U}]$, task i can either be completely executed before the start of the break or start after the end of the break. In the context of a continuous-time model (Castro et al. 2014a), it can be formulated as an exclusive disjunction featuring starting variable Ts_i and duration parameter p_i. When using a discrete-time model, break periods help to reduce model size by restricting the domain of task extent 0-1 variables $N_{i,t}$. In the example in Figure 2, the break lasts for 5 time intervals, including slots $\{9, \dots, 13\}$ (in red). Noting that durations p_i in minutes or hours are rounded up to multiples of δ (parameter specifying the length of every slot in the uniform grid), $\tau_i = \lceil p_i/\delta \rceil$, since the task lasts 5 slots (in grey), it can only start between slots 1 ($N_{i,1} = 1$) and 4 ($N_{i,4} = 1$), if the plan is to end before the break, or at slot 14, if started after the break, i.e. $T_i \in \{1, \dots, 4, 14\}$.

The goal of preemptive scheduling is to reduce idle times by allowing part of the task to be executed before the break and part after the break, see middle of Figure 3. Since the location of break periods is known a priori, one can easily determine the duration of a task $\bar{\tau}_{i,t}$ as a function of its starting point t. In the alternatives illustrated in Figure 3, the duration of the task changes between 5 (non-preemption duration), 10 (one interruption) and 12 (two breaks). The domain of the task is thus wider when allowing for preemption.

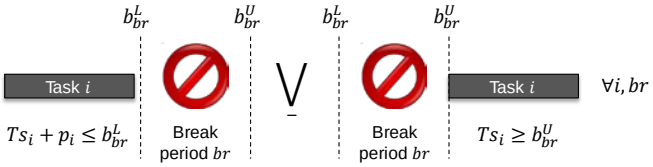

Figure 1. In non-preemptive scheduling, a task can either end-before or start-after a break.

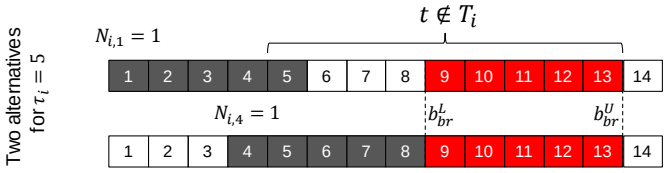

Figure 2. In non-preemptive scheduling, break periods allow reducing domain of processing tasks.

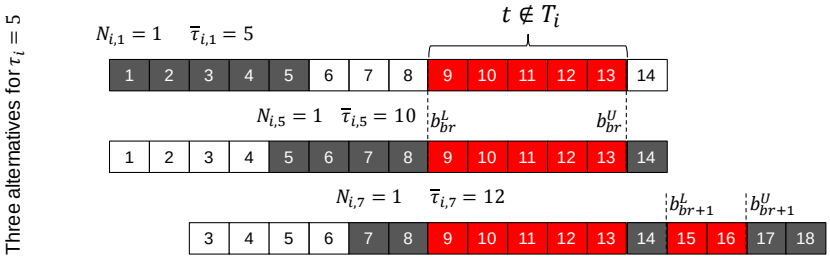

Figure 3. In preemptive scheduling, a task can be split over multiple periods.

3. New RTN discrete-time formulation

The remaining changes needed to extend the RTN discrete-time model to handle preemptive tasks occur at the level of the structural parameters and excess resource balances. Representing $\mu_{r,i,\theta}$ the amount of resource r consumed (-) or produced (+) by non-preemptive task i at a time θ relative to the start of the task, its preemptive counterpart $\bar{\mu}_{r,i,t,\theta}$ gains one index to identify the starting interval t. With the new structural parameters, resource consumption during task interruption can differ from normal execution. An example would be a residual energy consumption to compensate for heat losses during the break, which in our illustrative case study was set to 5%.

Considering that non-preemptive tasks are a special case, we simply need to make $\bar{\tau}_{i,t} = \tau_i \; \forall t \in T_i$, the excess resource balances of the new RTN formulation are given by Eq. (1). It features non-negative excess resource variables $R_{r,t}$ and parameters $\pi_{r,t}$, which account for the interaction with system boundaries (e.g. electricity purchase). Note that Eq. (1) is all that is needed to model preemption.

$$R_{r,t} = R_r^0|_{t=1} + R_{r,t-1} + \pi_{r,t} + \sum_{i'} \sum_{0 \le \theta \le \bar{\tau}_{i,t}: t-\theta \in T_i} \bar{\mu}_{r,i,t-\theta,\theta} N_{i,t-\theta} \; \forall r, t \tag{1}$$

4. Alternatives to minimizing makespan

Makespan minimization is perhaps the most difficult objective function in scheduling because it leads to poor linear programming (LP) relaxations of the mixed-integer linear programming (MILP) models. This is true for both discrete- and continuous-time formulations but it is more noticeable in the former for two reasons: (i) discrete-time formulations are very tight for cost-based objective functions (integrality gap very close to zero); (ii) problem sizes can be orders of magnitude larger, meaning more nodes to explore in the branch-and-bound tree, which is often translated into a poor computational performance. It is thus worth to revisit the alternatives for makespan minimization and evaluate their performance in the new preemptive tasks environment.

4.1. Most popular (option 1)

Makespan (MS) is defined as the maximum finishing time Tf_i over all tasks i, see Eq. (2). Assuming that all tasks are executed at most once during the time horizon, it can be expressed as a function of binary variables $N_{i,t}$ and parameters ft_t that define the position of slot t in minutes or hours. The standard way to minimize makespan is then to consider the linearization of the maximum function, Eq. (3). In cases where multiple instances of a task can be executed over the time horizon, one just needs to move index t from the domain of the summation in Eq. (3) to the inequality domain. Note that in a sequential facility, it suffices to include the subset of tasks linked to the last processing stage.

$$MS = \max_i Tf_i = \max_i (Ts_i + p_i) = \max_i (\textstyle\sum_t N_{i,t} \cdot ft_{t+\bar{\tau}_{i,t}}) \tag{2}$$

$$MS \ge \textstyle\sum_t N_{i,t} \cdot ft_{t+\bar{\tau}_{i,t}} \; \forall i \tag{3}$$

4.2. Defining a last task (option 2)

The second option was used by Castro et al. (2002) for an industrial problem with a superstructure with a task that was to be executed exactly once and was the last in the production sequence. In general, we can define an instantaneous last task i^* as one that consumes all given product demand. This is represented in Figure 4 for the case of a single batch per product (handled as unary resources). The makespan is then given by Eq. (4).

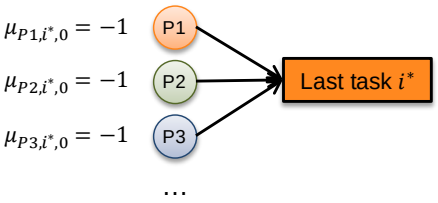

Figure 4. Structural parameters of the RTN associated to the definition of a last task.

$$MS = \sum_t N_{i^*,t} \cdot ft_t \quad (4)$$

4.3. Algorithmic search (option 3)

Maravelias and Grossmann (2003) proposed a very efficient iterative procedure for makespan minimization. Starting with a short time horizon H, insufficient to meet the demand, the problem is solved considering an objective function that is very sensitive to the schedule, e.g. total electricity cost. The solver will find very quickly that the problem is infeasible and so the next step is to keep adding one more time slot to the discrete-time grid per iteration until the problem becomes feasible. The number of slots then gives MS. The solution time per iteration will grow as we approach the optimal makespan (check chart in Castro et al., 2014b) and we want to start close enough. We thus propose to use the LP relaxation from option 2 (tighter than option 1), to set the initial number of slots. The drawback of option 3 is that the optimal solution is the first and lone feasible solution returned by the search procedure and is found only at the very end.

5. Computational results

We consider the demand side management problem of a steel plant with day-ahead, hourly electricity prices. The system and electricity price profile are taken from Castro et al. (2013). Note that to simplify the model, we allow idle times in the continuous caster units when handling heats belonging to the same group. The new RTN discrete-time scheduling formulation was implemented in GAMS 24.8.3 and solved with CPLEX 12.7 running in parallel deterministic mode using up to eight threads. The termination criteria were either a relative optimality tolerance of 10^{-6} or a maximum wall time limit of 7200 CPUs. The hardware consisted of a Windows 10, 64-bit desktop with an Intel i7-4790 (3.6 GHz) processor and 8 GB of RAM.

Table 1 presents the result for total cost minimization when considering breaks between [450, 510] and [930,990] (min). As expected, the wider domain of preemptive tasks, reflected in the larger number of discrete variables $N_{i,t}$, leads to a cost reduction. The difference to the non-preemptive case decreases from €16,253 (344,818-328.549) for time grids with δ=15 min slots, to a mere €147 for the more accurate δ=5 min. In the latter case, the optimal solution in Figure 5 features five tasks in preemptive mode. The computational time increases as δ decreases, as expected, but interestingly, only increases by a factor of two when considering preemptive tasks. Notice also that the formulation is very tight.

Table 2 and Table 3 present the results for makespan minimization. It can be seen that the standard way to minimize the makespan (option 1) is the worst since it can be as much as 5 times slower than the others and it returned a suboptimal solution of 1300 min for δ=5 min. The optimal solution of 1290 (Table 3) features 10 tasks being split over the two breaks to save 20 min of production time compared to the non-preemptive case

(makespan=1310 min, see Table 2). Option 2 was better for the non-preemptive mode, while option 3 prevailed otherwise. Overall, the impact of preemptive tasks in solution quality and computational time is similar for makespan and cost minimization.

Table 1. Computational statistics for total cost minimization

	Non-preemptive tasks				Preemptive tasks			
δ (min)	Discrete variables	RMIP (k€)	MIP (k€)	CPUs	Discrete variables	RMIP (k€)	MIP (k€)	CPUs
15	6012	344.817	344.818	4.96	6728	328.549	328.565	10.4
10	9042	314.691	314.844	22.1	10156	313.853	313.865	14.2
5	18072	310.367	310.436	3177	20400	310.271	310.289	6427

Table 2. Computational statistics for total makespan minimization (non-preemption)

		Option 1		Option 2		Option 3
δ (min)	Makespan (min)	RMIP (min)	CPUs	RMIP (min)	CPUs	CPUs
15	1440	872.15	27.4	937.81	**27.0**	65.0
10	1340	818.35	245	858.77	217	**151**
5	1310	808.11	2139	843.37	**666**	1026

Table 3. Computational statistics for total makespan minimization (preemption)

		Option 1		Option 2		Option 3
δ (min)	Makespan (min)	RMIP (min)	CPUs	RMIP (min)	CPUs	CPUs
15	1410	858.98	**86.5**	878.86	120	100
10	1340	816.16	1286	832.42	562	**260**
5	1290	789.44	7200	805.10	3535	**2951**

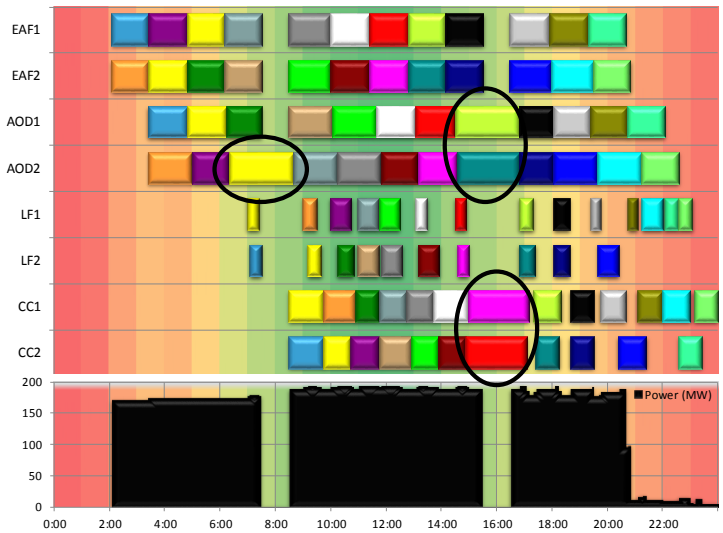

Figure 5. Optimal solution when minimizing electricity cost with δ=5 min time slots.

6. Conclusions

Whenever there is insufficient time to complete the execution of a processing task before a planned break, schedulers may be able to save time by splitting the execution of the task over the periods before and after the break. This preemptive mode of operation has mostly been neglected by scheduling formulations in process systems engineering. In this work, we have expanded a well-known and powerful discrete-time formulation to preemptive tasks. It involves acting at the level of structural parameters generation for the RTN, while keeping the same set of model variables and constraints. The benefits of considering preemption to improve solution quality where illustrated using an industrial case study from the literature, involving the demand side management of a steel plant. The results have also shown that while preemption makes the problem more difficult to solve, the computational time is of the same order of magnitude.

References

P.M. Castro, I.E. Grossmann, P. Veldhuizen, D. Esplin, 2014a, Optimal Maintenance Scheduling of a Gas Engine Power Plant using Generalized Disjunctive Programming, AIChE J., 60, 2083.

P. Castro, H. Matos, A.P.F.D. Barbosa-Póvoa, 2002, Dynamic Modelling and Scheduling of an Industrial Batch System, Comp. Chem. Eng. 26, 671.

P.M. Castro, D. Rodrigues, H.A. Matos, 2014b, Cyclic Scheduling of Pulp Digesters with Integrated Heating Tasks, Ind. Eng. Chem. Res. 53, 17098.

P.M. Castro, L. Sun, I. Harjunkoski, 2013, Resource−Task Network Formulations for Industrial Demand Side Management of a Steel Plant. Ind. Eng. Chem. Res. 52, 13046.

I. Grossmann, 2005, Enterprise-wide Optimization: A New Frontier in Process Systems Engineering, AIChE J., 51, 1846-1857.

I. Harjunkoski, R. Bauer, 2014, Sharing Data for Production Scheduling using the ISA-95 Standard, Frontiers in Energy Research, doi: 10.3389/fenrg.2014.00044.

I. Harjunkoski, C. Maravelias, P. Bongers, P.M. Castro, S. Engell, I. Grossmann, J. Hooker, C. Méndez, G. Sand, J. Wassick, Scope for Industrial Applications of Production Scheduling Models and Solution Methods, Comp. Chem. Eng. 62, 161.

E. Kondili, C.C. Pantelides, R.W.H. Sargent, 1993, A General Algorithm for Short-term Scheduling of Batch Operations – I. MILP Formulation. Comp. Chem. Eng. 2, 211.

H. Lee, C.T. Maravelias, 2017, Discrete-time Mixed-integer Programming Models for Short-term Scheduling in Multipurpose Environments. Comp. Chem. Eng. 107, 171-183.

C.T. Maravelias, I.E. Grossmann, 2003, Minimization of the Makespan with a Discrete-Time State−Task Network Formulation. Ind. Eng. Chem. Res. 42, 6252.

C.C. Pantelides, 1994, Unified Frameworks for the Optimal Process Planning and Scheduling, In Proceedings of the Second Conferencce on Foundations on Computer Aided Operations, Cache Publications, New York, 253.

V.V. Peteghem, M. Vanhoucke, 2010, A Genetic Algorithm for the Preemptive and Non-Preemptive Multi-mode Resource-Constrained Project Scheduling Problem, European Journal of Operational Research, 201,409-418.